Discriminative Hierarchical K-Means Tree for Large-Scale Image Classification

Shizhi Chen, Xiaodong Yang, and Yingli Tian, *Senior Member, IEEE*

***Abstract*— A key challenge in large-scale image classification is how to achieve efficiency in terms of both computation and memory without compromising classification accuracy. The learning-based classifiers achieve the state-of-the-art accuracies, but have been criticized for the computational complexity that grows linearly with the number of classes. The non-parametric Nearest-Neighbor (NN) based classifiers naturally handle large numbers of categories, but incur prohibitively expensive computation and memory costs. In this paper, we present a novel classification scheme, i.e., Discriminative Hierarchical K-Means Tree (D-HKTree), which combines the advantages of both learning-based and NN-based classifiers. The complexity of the D-HKTree only grows sub-linearly with the number of categories, which is much better than the recent hierarchical Support Vector Machines (SVM) based methods. The memory requirement is the order of magnitude less than the recent Naïve-Bayesian Nearest-Neighbor (NBNN) based approaches. The proposed D-HKTree classification scheme is evaluated on several challenging benchmark databases and achieves the state-of-the-art accuracies, while with significantly lower computation cost and memory requirement.**

Index Terms—Hierarchical K-Means Tree (HKTree), Large Scale, Image Classification, Naïve Bayesian Nearest Neighbor (NBNN), and Support Vector Machine (SVM).

I. INTRODUCTION

IMAGE classification has remained one of the most fundamental problems in computer vision, with various applications, such as surveillance and augmented reality *etc*. At the same time, the available image data has increased dramatically due to the widely accessible Internet, e.g., YouTube, Facebook. Hence, the large-scale image classification has attracted significant research efforts, especially when the number of classes scales up [6, 10, 22]. One of the key challenges in the large-scale image classification is to achieve efficiency in term of both computation and memory without compromising classification accuracy.

Manuscript received Oct. 7th 2013. This work was supported in part by NSF grants EFRI-1137172 and IIS-1400802, FHWA grant DTFH61-12-H-00002, and ONR grant N000141310450.

Shizhi Chen is with Naval Undersea Warfare Center (NUWC), Newport, RI 02841 USA (e-mail: shizhi.chen@navy.mil). The work was conducted when he was with City College, City University of New York.

Xiaodong Yang is with City College, City University of New York, New York, NY 10031 USA (e-mail: xyang02@ccny.cuny.edu).

YingLi Tian is with City College and Graduate Center, City University of New York, New York, NY 10031 USA (phone: 212-650-7046; fax: 212-650-8249; e-mail: ytian@ccny.cuny.edu).

The learning based classifiers, including Support Vector Machine (SVM) [4, 5, 12], Random Forest [26] and Adaboost [21], have demonstrated excellent performance for the image classification. However, these learning based classifiers usually do not scale well on the increasing number of image categories. For example, the one-versus-all linear SVM, one of the most popular classifiers for the large-scale image classification, has the computation complexity linearly increasing with the number of image categories. To improve the efficiency of the linear SVM, Griffin *et al*. [11] built class taxonomies by measuring affinity between a pair of classes with a confusion matrix. In order to achieve a better tradeoff between accuracy and efficiency, several papers [7, 15] utilized the relaxed hierarchical SVM, which postpones the decision for some confusing classes in a hierarchical decision structure. However, all of these approaches improve classification efficiency through compromising classification accuracy to some extent.

On the other hand, non-parametric Nearest Neighbor (NN) based classifiers require no training and can naturally handle large numbers of classes [2, 16, 25, 28]. However, they have to retain all the training examples in the testing phase, which becomes infeasible even on moderate-scale image classification, because of the expensive memory and computation cost.

Compared to learning-based classifiers, accuracies of NN-based classifiers are normally much lower, which also limits their applications for image classification. The Naïve-Bayesian Nearest-Neighbor (NBNN) [2] improves the classification accuracy of NN-based classifiers by avoiding vector quantization and utilizing the image-to-class distance metric. However, the computation cost is still linearly proportional to the number of classes, even after using some approximate NN techniques [18]. Local NBNN [16] improves the computation cost of the NBNN by only updating the classes found in a local neighborhood. Hence, the complexity only grows logarithmically with the number of categories. Nevertheless, the computation cost can still be expensive when the training data is large and exceeds the memory capacity. Both NBNN and local NBNN need to retain all training samples.

Despite these efforts [1, 2, 16, 20, 24] have been made to improve the performance of NN-based classifiers, very few work has tried to extend the NN-based classifiers to large-scale image classification by reducing both computation and memory costs.

Some researchers take advantages of the complementarities between classifiers of Nearest-Neighbor and SVM [20, 24, 27]. Tuytellaars *et al*. [20] integrated both

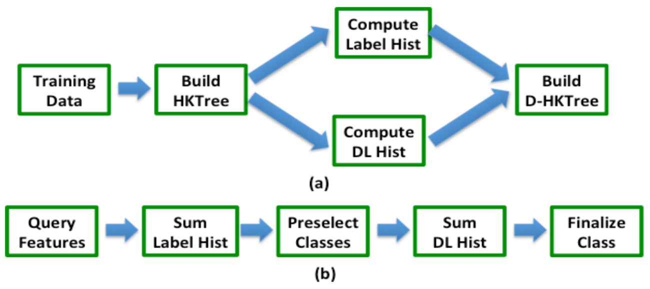


Figure 1: Overview of the D-HKTree for (a) training; (b) testing;

NBNN and SVM with Bag-of-Words (BOW) representation into a Multiple Kernel Learning (MKL) framework [8]. In order to reduce the expensive computation cost of the NBNN kernel, they had to down-sample the query feature vectors in a testing image, which adversely affect classification results.

In this paper, we first derive and formulate a novel nearest neighbor method, the Labeled Hierarchical K-Means Tree (L-HKTree), which can dramatically reduce the computation and memory complexity as compared with other NN-based classifiers. By pre-computing label statistics of nearest neighbors for training data, the L-HKTree can infer the class label of a query image by simply looking up the label statistics.

Built on the efficiency of the proposed NN-based classifier, we further propose a Discriminative Hierarchical K-Means Tree (D-HKTree) which takes advantages of both learning-based and NN-based methods. The computational complexity of the D-HKTree only grows sub-linearly with the number of image categories, while still achieves the state-of-the-art accuracy on several challenge datasets [6, 10, 22].

II. Discriminative Hierarchical K-Means Tree

The proposed Discriminative Hierarchical K-Means Tree (D-HKTree) extends the NN-based classification scheme to large-scale image classification. In this section, we describe the detailed theoretical derivations and computational procedures of the L-HKTree and the D-HKTree.

A. *Algorithm Overview*

Fig. 1(a) and 1(b) show the flowcharts of the D-HKTree framework during training and testing phases respectively. In the training step, we construct a Hierarchical K-Means Tree (HKTree) [19] from all the training feature vectors. Then we compute both label histogram and Discriminatively Learned (DL) histogram in parallel. The label histogram pre-computes the label statistics of the nearest neighbors of the query feature vectors. After the label histograms populate leaf nodes of the HKTree, we construct the L-HKTree. The DL histogram captures the weights of a linear classifier for each class. The DL histogram is stored in the non-leaf nodes of the HKTree at a selected level to form the D-HKTree. Feature vectors of a query image first find the leaf nodes of the D-HKTree, and sum their label histograms. Based on the accumulated label histogram, we preselect the most confident categories of the image. Finally the query feature vectors propagate back to the non-leaf nodes of the D-HKTree, and sum DL histograms only over the preselected classes. The

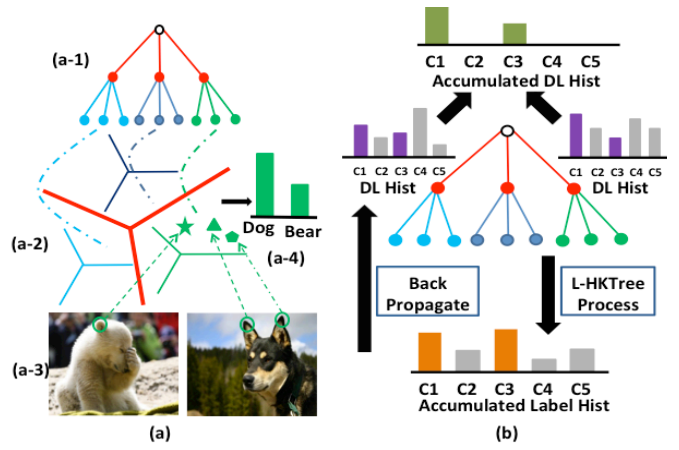


Figure 2: (a) The construction of the Labeled Hierarchical K-means Tree (L-HKTree); (b) The prediction step of the D-HKTree for the image classification.

final class label is the most confident class label of the pre-selected classes in the accumulated DL histogram.

B. *Labeled Hierarchical K-Means Tree*

The main structure of the D-HKTree is built upon the L-HKTree. In this section, we describe the detailed procedures in deriving and building the L-HKTree.

Same as in the NBNN algorithm, we make two assumptions, i.e., uniform prior over all class labels and the independence of the query feature vectors d_i in a testing image Q [2]. The predicted label can be obtained by Eq. (1).

$$\hat{C} = \underset{C}{\operatorname{argmax}} \sum_{\mathrm{i}} \log p(d_i|C) \ . \tag{1}$$

We explicitly model unbalanced data over class labels in the Parzen window estimator. The total number of training feature vectors in class C is L_C.

$$\log p(d_i|C) = \frac{1}{L_C} \sum_{j=1}^{L_C} K(d_i - d_j^C) \ . \tag{2}$$

Instead of using the Gaussian kernel for K as in the local NBNN, we use a uniform kernel with the bandwidth of r_i.

$$K\left(d_i - d_j^C\right) = B_f \ U\left(1 - \frac{\left\|d_i - d_j^C\right\|}{r_i}\right) \ , \tag{3}$$

where the bandwidth $r_i > 0$. B_f is a positive constant and U is a unit step function, which is 1 if the Euclidean distance between the training feature vector d_j^C and the query feature vector d_i is less than the bandwidth r_i. Otherwise the step function is 0. If substituting Eq. (3) to Eq. (2), we have

$$\log p(d_i|C) = \frac{B_f}{L_C} \sum_{j=1}^{L_C} U\left(1 - \frac{\left\|d_i - d_j^C\right\|}{r_i}\right) . \tag{4}$$

The summation term in Eq. (4) denotes the total number of training feature vectors in class C, which have Euclidean distance smaller than r_i away from the query feature vector d_i as illustrated in Fig. 3(a). The center of the circle is at the query feature vector d_i with the radius equals to the bandwidth of r_i. The summation term in Eq. (4) for the triangle class is the total number of triangle training feature vectors falling within the circle, i.e., 2 in Fig. 3(a). Similarly, the summation terms for the pentagon class and the star class are 3 and 1, respectively. As illustrated in Fig. 3(b), we further approximate the unit step function U with the feature space boundary defined by the leaf node f_i, in which the query feature vector d_i falls.

$$U\left(1-\frac{\left\|d_i-d_j^C\right\|}{r_i}\right) \cong \delta\left(f_i, \mathrm{LEAF}\left(d_j^C\right)\right) \ , \tag{5}$$

where $\mathrm{LEAF}(d_j^C)$ is the nearest neighboring leaf node of the training feature vector d_j^C; δ equals to 1 when feature vectors d_i and d_j^C fall within the same feature space partition defined by the leaf node f_i. Otherwise, it is equal to 0. If we substitute Eq. (5) to Eq. (4), it becomes

$$\log p(d_i|C) = \frac{B_f}{L_C} \sum_{j=1}^{L_C} \delta\left(f_i, \mathrm{LEAF}\left(d_j^C\right)\right) . \tag{6}$$

Note that the right-hand side of the Eq. (6) does not depend on the query feature vector d_i except that f_i is the leaf node of d_i. Therefore, we can pre-compute the right-hand side of Eq. (6) for each class label C, and store the results as the label histogram associated with the leaf node f_i, denoted as $LH(f_i, C)$.

The summation term in Eq. (6) is the number of training feature vectors from category C, which fall within the feature space partition of the leaf node f_i. L_C corresponds to the total number of training feature vectors in category C. B_f is a L1-Norm constant. Substituting Eq. (6) to Eq. (1), we have the L-HKTree classification rule.

$$\hat{C} = \mathrm{argmax}_C \sum_{\mathrm{i}} \left(\frac{B_f}{L_C} \sum_{j=1}^{L_C} \delta\left(f_i, \mathrm{LEAF}\left(d_j^C\right)\right)\right)$$
$$= \mathrm{argmax}_C \sum_i LH(\mathrm{LEAF}(d_i), C) \ . \tag{7}$$

As demonstrated in the derivation of the L-HKTree, we do not need to compute the pair-wise distance of the query feature vector d_i with each training feature vector d_j^C online. Instead, we only employ the label histogram associated with the leaf node where d_i falls within. The label histograms of the leaf nodes are summed up together to predict the class label of an image as shown in Eq. (7). Hence, L-HKTree does not retain any training feature vector. This translates to a significant saving in the memory, which allows us to extend this NN-based classifier to a large-scale classification task.

Furthermore, the computational complexity of L-HKTree during testing is independent of the number of classes, which is a very attractive property for image classification on large-scale datasets with large number of categories. The label histograms in the leaf nodes can be very sparse, since the number of leaf nodes in L-HKTree is exponentially increasing with the number of levels.

To build the L-HKTree, we first construct a Hierarchical K-Means Tree (HKTree) [19] from training feature vectors. Fig. 2(a-1) illustrates a two-level HKTree with three branches. The corresponding feature space partition of each leaf node projected on the two-dimensional space is shown in Fig. 2(a-2). Training feature vectors are clustered at the first level by the K-Means with the number of centers equal to the tree branches K. Then at the successive levels, the HKTree will continue splitting feature vectors in each branch by the K-Means until reaching level L.

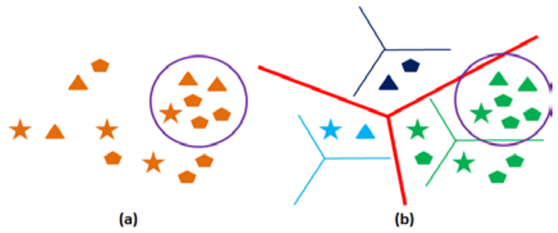


Figure 3: Illustration of unit step function. (a) The circle illustrates the unit step function in Eq. (4); (b) Approximate the unit step function of the circle with the feature space partition by one of the leaf nodes in the HKTree.

We modify the original hierarchical K-Means tree to automatically reduce the number of branches of a non-leaf node if the average number of training feature vectors arriving at its children nodes is below a threshold N. N is related to minimum number of nearest neighbor training feature vectors falling in leaf nodes, similar to the K nearest neighbors for the local NBNN [16]. Best performance is achieved in our experiments when N is set as 15.

The leaf nodes in the L-HKTree have defined their corresponding feature space boundaries, as illustrated in Fig. 2(a-2). Any feature vector arriving at a leaf node can be considered as a nearest neighbor of this leaf node. Fig. 2(a-3) illustrates that the "ear" feature vectors of dog and bear class arrive at the same leaf node of the L-HKTree. The label histogram associated with each leaf node summarizes the number of nearest neighbor training feature vectors over class labels. Fig. 2(a-4) illustrates the label histogram over dog and bear categories. Intuitively, the more nearest neighbor feature vectors are from the class label C in the label histogram of the leaf node f, the smaller distance is between the leaf node f and the class label C. Therefore, we can have another interpretation of the label histogram as the inverse distance from leaf node f to different classes.

The max-pooling [13] used in the Bag of Words (BOW) framework achieves excellent performance by selecting feature vectors, which is the nearest neighbor feature vector of a visual word in the codebook. We also implement similar filtering techniques in the L-HKTree. The non-leaf nodes, which store the DL histograms in the D-HKTree, are used as filters. Only query feature vectors which are the nearest neighbors of one of the non-leaf nodes are allowed to continue down the L-HKTree and participate in the accumulation process of label histograms.

C. *Discriminative Hierarchical K-Means Tree*

We integrate a learning-based classifier into the L-HKTree framework to form the Discriminative Hierarchical K-Means Tree (D-HKTree).

The learning step of the D-HKTree involves the computation of the Discriminatively Learned (DL) histograms, which are stored in the non-leaf nodes at a selected level of the L-HKTree. The DL histograms H capture the weights (*i.e.*, the normal vectors) of linear classifiers over different classes. We first compute the Bag of Words (BOW) representation [9] by treating the non-leaf nodes at the selected level as visual words. Then we train a linear classifier on the BOW for each class c. The value w_i^c

of the normal vector for class c is stored in the DL histogram H_c at i^{th} visual word (*i.e.*, non-leaf node). We choose the one-versus-all linear SVM as the linear classifier.

Fig. 2(b) demonstrates the classification step of the proposed D-HKTree. Query feature vectors in a testing image arrive at their nearest neighbor leaf nodes of the L-HKTree. The corresponding label histograms are summed up into the accumulated label histogram, where top P performance class labels are selected. After the L-HKTree process, query feature vectors propagate back to their parent non-leaf nodes at the selected level, e.g., the red nodes in Fig. 2 (b). The DL histograms are weighted with the distance from their parent non-leaf nodes. Then the weighted DL histograms over the selected P class labels are summed up into the accumulated DL histogram at the top of the figure. Finally, we select the class label with the highest score from the accumulated DL histogram. Since the L-HKTree does not store any training data, the memory cost of the D-HKTree is dramatically reduced as compared with other NN-based methods [2, 16].

The number of the top candidate classes P is automatically selected for different testing images by using a fixed cumulative confidence level (CCL). After we sort the accumulated label histogram over the classes in the descending order, P is determined as the minimum number of classes while the cumulative probability of the accumulated label histogram is greater than the specified CCL value. In our experiments, we set the value of CCL to 0.2, which achieves the best tradeoff between the computation complexity and accuracy. By only selecting the top P classes in the forward L-HKTree process, the D-HKTree is able to achieve the computation complexity that grows sub-linearly with the number of classes, which is much better than the learning-based classifiers.

III. Experiments

We evaluate our proposed frameworks on several object and scene recognition datasets including Caltech 101 [6], Caltech 256 [10], and SUN datasets [22]. The proposed D-HKTree significantly outperforms all previous NN-based classifiers in terms of classification accuracy, computation cost, and memory requirement. The relative computational complexity of the D-HKTree is also significantly improved compared to the state-of-the-art learning-based classifiers. Experimental results demonstrate that the D-HKTree can scale very well to large-scale datasets with large numbers of categories.

A. *Experimental Setup*

We employ the dense SIFT feature vectors [14] augmented by x and y coordinates throughout our experiments, following the approach in the local NBNN [16]. The dense SIFT feature vectors are extracted by sampling every 3 pixels at 3 scales, and removing low contrast points. The two spatial dimensions in the augmented feature vector are weighted by 1.6 in Caltech 101, 0.75 in Caltech 256 and SUN, as recommended in [16, 17].

To achieve high accuracy, a large codebook size of visual words has been suggested for a linear classifier [3, 17]. We have experimentally verified that the best performances are achieved when the codebook size is 65K for the Caltech 101 database and 130K for the Caltech 256 and SUN databases. To simplify the experiment, the L-HKTree has 2 levels.

Table 1: Comparison to NN-based classifiers on accuracy and speed

Dataset	Caltech 101		Caltech 256		SUN	
Method	accuracy (%)	speed (s/im)	accuracy (%)	speed (s/im)	accuracy (%)	speed (s/im)
1-NN	-	-	7.6±0.7* [10]	-	13* [22]	-
NBNN [2]	70.4*	23.29	37*	450.5	-	-
NBNN [20]	65.5±1.0*	-	-	-	-	-
Local NBNN	71.9 ±0.6*	2.89	40.1±0.1*	112.4	-	-
D-HKTree	**77.6±0.66**	**1.34**	**45.5±0.58**	**3.7**	**35.7±0.18**	**1.89**

In order to facilitate a fair comparison, we follow the evaluation conventions, i.e., 30 images per category are used as training data and 15 images per category are used as testing data in the Caltech 101 and the Caltech 256 datasets. We repeat the experiments 10 times with random selection of non-overlapping training and testing data. The average accuracy with the standard deviation is reported in the paper. As for the SUN dataset, we use exactly the same training and testing splitting as in [22], i.e., 50 images for both training and testing sets.

For the cited classification accuracies in this paper, most results on the Caltech 101 and the Caltech 256 are based on 30 training images per class unless otherwise noted. For the SUN dataset, they are all based on 50 training images per class. Most of the cited results are based on the spatial variants of dense SIFT or Histogram of Gradient (HOG), even though some of them use multiple feature types.

B. *Comparisons to NN-based Classifiers*

In this paper, the "*" sign after a number indicates that the number is directly quoted from the original papers. The "-" sign in the tables indicates that the data is not available.

Table 1 shows the comparison of the D-HKTree with other NN-based classifiers on both classification accuracy and computation cost.

The D-HKTree has achieved the highest classification accuracies, i.e., 77.6% and 45.5%, on the Caltech 101 and the Caltech 256 respectively, which are 5% higher than the state-of-the-art NN-based method, i.e., local NBNN [16]. We have verified the reported local NBNN accuracy by running the source code provided by the paper [16] on the Caltech 101. Our proposed D-HKTree achieves 35.7% accuracy on the SUN dataset. To the best of our knowledge, this is the highest accuracy on this dataset using a single feature type. We also quote the classical 1-NN classifier results on the Caltech 256 and the SUN datasets for the comparison in the Table 1. The 1-NN classifier is a correlation classifier in the feature space of pixel intensities of a resized image [10]. The 1-NN results reported in [22] are based on multiple feature types. If using a single feature type, the results may even worse.

The testing speed of the D-HKTree is significantly faster than the conventional NN-based classifiers, especially on larger datasets. To evaluate the testing speed, we run the source code provided by [16] with the recommended parameters for the NBNN and the local NBNN methods. If we use 30 training images per category on the Caltech 256 dataset, the testing speed of the D-HKTree is 30 times faster than the local NBNN, and 120 times faster than the NBNN.

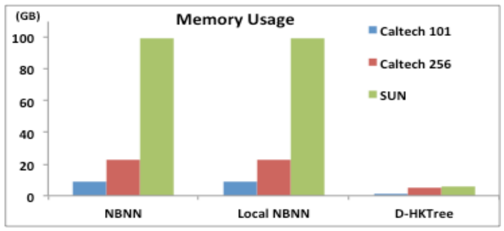


Figure 4: Comparison of the memory usage of different NN-based classifiers as the scale of the dataset increases.

Table 2: Comparison of accuracy to learning based classifiers on (a) Caltech 101 and Caltech 256 datasets; (b) SUN dataset.

Method	Caltech 101 (%)	Caltech 256 (%)
Orig SPM	64.6±0.8* [12]	34.1±0.2* [10]
SC SPM [23]	73.2±0.54*	34±0.35*
SLC	**78.39±0.59**	**45.71±0.59**
KC [9]	64.1±1.5*	27.2±0.4*
D-HKTree	**77.6±0.66**	**45.5±0.58**

(a)

Method	SUN (%)
Orig SPM [22]	21.5*
Spatial HOG[22]	27.2*
Spatial HOG[7]	27*
D-HKTree	**35.7±0.18**

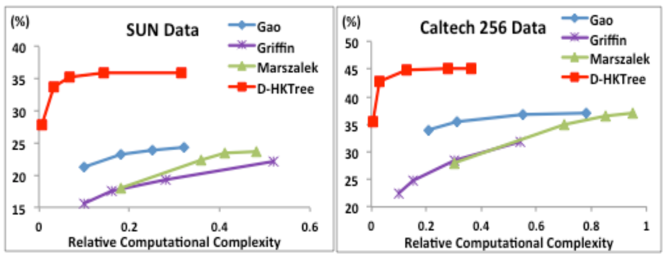
(b)

As shown in Table 1, the local NBNN is 10 times faster than the NBNN on the Caltech 101 dataset, but only 4 times faster on the Caltech 256, which is different from the results reported in [16]. This is because that the training data in the Caltech 256 (~59 GB) have exceeded the memory capacity (~48 GB) of the computer we used. The test speed of the conventional NN-based classifiers on the SUN database is difficult to evaluate, because it requires approximately 100 GB of memory to store all training data, which far exceed the memory capacity of our computers. On the other hand, the computation cost increment is significantly lower for the D-HKTree as the dataset changes from the Caltech 101 to the Caltech 256. We observe that the computation cost of the SUN dataset is even lower than that of the Caltech 256, i.e., 1.9 second per image versus 3.7 second per image. This is due to the structure difference of the L-HKTrees built for the two datasets, since the major computation costs of D-HKTree is from the pre-classification of L-HKTree.

Fig. 4 compares the memory requirements of different NN-based classifiers as the scale of dataset increases. Since the memory usage of the conventional NN-based classifiers is directly related to the size of training data, we can estimate their memory usages according to the training data size. As for the D-HKTree, the memory usage is estimated from the size of the L-HKTree. As shown in this figure, the memory usage of the D-HKTree is pretty stable and only increases slightly from the Caltech 101 to the Caltech 256, then the SUN dataset. However, the memory consumptions of the NBNN and the local NBNN grow significantly as the dataset scaling up. For example, the memory requirement is around 100 GB for both the NBNN and the local NBNN in SUN dataset, while the memory usage of the D-HKTree is only 6GB.

C. *Comparisons to Learning-based Classifiers*

Table 2 demonstrates the comparisons of the classification accuracy between the D-HKTree and the state-of-the-art learning-based methods. The D-HKTree outperforms most learning-based methods and achieves comparable



(a) (b)

Figure 5: Comparison of the tradeoff between accuracy and relative computational complexity to hierarchical SVM-based methods for large scale data, i.e., Gao [7], Griffin [11], Marszalek [15], on (a) SUN dataset; (b) Caltech 256 dataset; Note that the results for other three methods are directly estimated from the plots in the paper [7].

performance to the recently proposed Spatially Local Coding (SLC) [17] on the Caltech 101 and the Caltech 256. The SLC method uses the linear SVM classifier on the Bag of Words (BOW) feature representation. However, the SLC needs to evaluate all class classifiers. Hence, it cannot scale well on the large-scale dataset with large number of categories.

As for the SUN dataset in Table 2(b), the D-HKTree significantly outperforms the current state-of-the-art with a single feature type by more than 8% in classification accuracy. Note that there are almost 400 image categories in the SUN dataset.

To further evaluate the scalability to large-scale image classification, we compare the D-HKTree with several hierarchical SVM-based classifiers [7, 10, 15] in Fig. 5. Note that 40 training images per class are used for the literature results reported in Fig. 5, while the D-HKTree only uses 30 training images per class. All of these hierarchical SVM-based classifiers attempt to improve the efficiency of one-versus-all linear SVM classifier, so that the complexity can grow sub-linearly with the number of categories. However, these classifiers have to sacrifice classification accuracy for the improvement on speed. We adopt the relative computational complexity (RC) measure introduced in [7] for our evaluation. In the case of a linear kernel, the relative complexity is the ratio between the number of categories evaluated and the total number of categories in the dataset.

The relative computational complexity of the D-HKTree can be tuned by varying the number of top selected class labels from the accumulated label histogram, or by the Cumulative Confidence Level (CCL). Although the computation cost of the L-HKTree is not reflected in this measure, this cost is independent of the number of categories. As demonstrated in Fig. 5, the D-HKTree dominates the classification accuracies on the Caltech 256 and the SUN datasets, especially when the relative computational complexity is low. For instance, at the relative computational complexity of 0.06 in Fig. 5(a), the D-HKTree achieves 35% on the SUN dataset, which is more than 10% higher than the best result reported in [7]. Similar results are shown on the Caltech 256 dataset in Fig. 5(b). We observe that the classification accuracy of the D-HKTree tends to saturate around the relative computational complexity of 0.1, which means that the D-HKTree is more effective to reduce the relative computational complexity and maintains a desirable accuracy.

Table 3: Comparison to the hybrid classifiers, i.e., SVM-KNN [24], and NBNN Kernel [20], on the Caltech 101 dataset.

Method	SVM-KNN	NBNN Kernel	NBNN Multi-kernel	D-HKTree
Accuracy (%)	66.2±0.5*	69.6±0.9*	75.2±1.2*	**77.6±0.66**

Table 4: The effect of cumulative confidence level (CCL) on the accuracy and the relative computational complexity (RC).

CCL	0.01	0.02	0.05	0.1	0.2	0.3
RC	0.00	0.01	0.03	0.05	0.11	0.17
Accuracy	66%	69%	75%	77%	79%	79%

D. *Comparisons to Hybrid Classifiers*

There are very few work [20, 24] on combining the learning-based and the NN-based classifiers to take the advantages of both types of classifier. Table 3 compares the D-HKTree with two other methods that hybrid both classifier types, i.e., SVM-KNN [24] and NBNN Kernel [20]. As shown in this table, the D-HKTree significantly outperforms the SVM-KNN and the NBNN Kernel by 11% and 8% in accuracy respectively. Note that the NBNN Multi-Kernel is actually combining the NBNN Kernel with other kernels of different feature types instead of a single feature type. Nevertheless, the D-HKTree still obtains the highest accuracy as shown in Table 3.

Finally, we evaluate the effect of the cumulative confidence level (CCL) on the performance of the D-HKTree including both relative computational complexities (RC) and accuracy. The experiment is conducted on the first set of training and testing data of the Caltech 101. The result is shown in Table 4. As the CCL increases, more classes are forwarded from the L-HKTree process to the discriminative classifiers stored in the DL histograms. Hence the RC increases as well as the classification accuracy. Based on Table 4, we observe that the CCL of 0.2 achieves a high accuracy while maintaining a low relative computational complexity of 0.11. Note that the RC is 0, when the CCL equals to 0.01. That indicates that the final class label is inferred by the L-HKTree only without propagating back to the non-leaf nodes, which stores the DL histograms.

IV. Conclusion

In this paper, we have proposed a novel classification scheme, i.e., D-HKTree, for large scale image classification. The D-HKTree combines the advantages of both learning-based and NN-based methods. It extends the ability of the NN-based classifiers to handle large-scale image classification with much lower computation cost and memory requirement, and achieves the state-of-the-art classification accuracies. Compared to NN-based methods, the D-HKTree significantly outperforms the NBNN and the local NBNN in classification accuracy, computation and memory costs. Compared to the learning-based methods, the D-HKTree largely improves the accuracy of the hierarchical SVM-based methods at much lower relative computational complexity. The D-HKTree also achieves comparable accuracy with the SLC method. Compared to previous hybrid methods, the D-HKTree obtains much better performance than the SVM-KNN and the NBNN Kernel.

References

[1] R. Behmo, P. Marcombes, A. Dalalyan, and V. Prinet, “Towards Optimal Naïve Bayes Nearest Neighbor”, European Conference of Computer Vision (ECCV), 2010.

[2] O. Boiman, E. Shechtman, and M. Irani, “In Defense of Nearest Neighbor Based Image Classification”, Computer Vision and Pattern Recognition (CVPR), 2008.

[3] Y. Boureau, F. Bach, Y. LeCun, and J. Ponce, “Learning Mid-Level Features For Recognition”, Computer Vision and Pattern Recognition (CVPR), 2010.

[4] C. Chang and C. Lin, “LIBSVM: a library for support vector machines”, ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011.

[5] C. Cortes and V. Vapnik, "Support-Vector Networks", Machine Learning, 20, 1995

[6] L. Fei-Fei, R. Fergus and P. Perona, “One-Shot learning of object categories”, IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), 2006.

[7] T. Gao and D. Koller, “Discriminative Learning of Relaxed Hierarchy for Large Scale Visual Recognition”, International Conference on Computer Vision (ICCV), 2011.

[8] M. Gonen and E. Alpaydm, “Multiple Kernel Learning Algorithms”, Journal of Machine Learning Research, 12(Jul):2011-2268, 2011.

[9] J. Gemert, C. Veenman, and A. Smeulders, J. Geusebroek, “Visual Word Ambiguity”, IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), 2010.

[10] G. Griffin, A. Holub, and P. Perona, “Caltech-256 object category dataset”, Technical Report 7694, California Institute of Technology, 2007.

[11] G. Griffin and P. Perona, “Learning and using taxonomies for fast visual categorization”, Computer Vision and Pattern Recognition (CVPR), 2008.

[12] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories”, Computer Vision and Pattern Recognition (CVPR), 2006.

[13] L. Liu, L. Wang, and X. Liu, “In Defense of Soft-assignment Coding”, International Conference on Computer Vision (ICCV), 2011.

[14] D. Lowe, “Distinctive Image Features from Scale Invariant Keypoints”, International Journal of Computer Vision, 2004.

[15] M. Marszalek and C. Schmid, “Constructing category hierarchies for visual recognition”, European Conference of Computer Vision (ECCV), 2008.

[16] S. McCann and D. Lowe, “Local Naïve Bayes Nearest Neighbor for Image Classification”, Computer Vision and Pattern Recognition (CVPR), 2012.

[17] S. McCann and D. Lowe, “Spatially Local Coding for Object Recognition”, Asian Conference on Computer Vision (ACCV), 2012.

[18] M. Muja, D. Lowe, “Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration”, International Conference on Computer vision Theory and Applications (VISAPP), 2009.

[19] D. Nister and H. Stewenius, “Scalable Recognition with a Vocabulary Tree”, Computer Vision and Pattern Recognition (CVPR), 2006.

[20] T. Tuytelaars, M. Fritz, K. Saenko, and T. Darrell, “The NBNN Kernel”, International Conference on Computer Vision (ICCV), 2011.

[21] Y. Freund and R. Schapire, “A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting”, Journal of Computer and System Sciences, 1997.

[22] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba, “SUN Database: Large-Scale Scene Recognition from Abbey to Zoo”, Computer Vision and Pattern Recognition (CVPR), 2010.

[23] J. Yang, K. Yu, Y. Gong, and T. Huang, “Linear Spatial Pyramid Matching Using Sparse Coding for Image Classification”, Computer Vision and Pattern Recognition (CVPR), 2009.

[24] H. Zhang, A. Berg, M. Maire, and J. Malik, “SVM-KNN: Discriminative Nearest Neighbor Classification for Visual Category Recognition”, Computer Vision and Pattern Recognition (CVPR), 2006.

[25] F. Angiulli and A. Astorino, “Scaling up Support Vector Machines using Nearest Neighbor Condensation”, IEEE Trans. Neural Networks, Vol. 21, 2010.

[26] L. Breiman, "Random Forests", Machine Learning, 2001.

[27] C. Domeniconi, D. Gunopulos, and J. Peng, “Large Margin Nearest Neighbor Classifiers”, IEEE Trans. Neural Network, Vol. 16, 2005.

[28] H. Fayed and A. Atiya, “A Novel Template Reduction Approach for the K-Nearest Neighbor Method”, IEEE Trans. Neural Networks, Vol. 20, 2009.